Subregular Morpho-Semantics
The Expressive Limits of Monomorphemic Quantifiers

Thomas Graf

Stony Brook University
mail@thomasgraf.net
http://thomasgraf.net

Rutgers University
December 15, 2017

You can get the slides here under “News”
Take-Home Message

- Supplement linguistic theory with computational perspective
- Typological gaps can be explained computationally.

Case Study: Morphosemantics of Quantifiers

A D-quantifier may have a monomorphemic realization only if its quantifier language is TSL.
Outline

1. TSL Patterns in Phonology, Morphology, and Syntax

2. TSL Morpho-Semantics
   - Quantifier Languages
   - All Monomorphemic Quantifiers are TSL
   - Tightening the Characterization

3. A Broader Program
Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity

1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?
Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity

1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?
Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity

1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?
Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity

1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?
TSL: Tier-Based Strictly Local

- All patterns described by markedness constraints that are
  - inviolable,
  - locally bounded,
  - formalized as $n$-grams.

- Non-local dependencies are **local over tiers**.
  (Goldsmith 1976)

- **Linguistic core idea:**
  Dependencies are local over the right structure.
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $\sim$ = word edge).

**Example: German**

\[
\begin{array}{c}
\text{\ } r \quad a \quad d \quad \text{\ }
\end{array}
\]

\[
\begin{array}{c}
\text{\ } z \quad \sim \quad \text{\ }
\text{\ } v \quad \sim \quad \text{\ }
\text{\ } d \quad \sim \quad \text{\ }
\end{array}
\]
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $\_ = \text{word edge}$).

### Example: German

<table>
<thead>
<tr>
<th>$_ r a d _$</th>
<th>$_ r a d$</th>
<th>$_ r a d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^*z____$</td>
<td>$^*v____$</td>
<td>$^*d____$</td>
</tr>
</tbody>
</table>
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have \( z\$ \) or \( v\$ \) or \( d\$ \) (where \( \$ = \) word edge).

Example: German

```
* $ ra d $  
* z$
* v$
* d$
```
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z\$ or $v\$ or $d\$ (where $\$ = word edge).

Example: German

```
* $\$ r a d \$
  * $z\$
  * $v\$
  * $d\$
```
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have \( z \) or \( v \) or \( d \) (where \( $ \) = word edge).

**Example: German**

```plaintext
*$ $ \text{rad}$ $*$
*$z$
*$v$
*$d$
```
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $\$ = word edge).

**Example: German**

```
* $ r a d $
```

```
* $z$
* $v$
* $d$
```
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $\$ = word edge).

Example: German

```
* $ r a d

* z$

* v$

* d$
```
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $=$ word edge).

Example: German

```
* $r a d$
* $z$
* $v$
* $d$
```

```
$r a t$
```

References
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $=$ word edge).

**Example: German**

\[
\begin{array}{c}
* \text{r a d} \\
* \text{z} \\
* \text{v} \\
* \text{d} \\
\hline
\text{r a t}
\end{array}
\]}
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z$ or $v$ or $d$ (where $\$ = word edge).

**Example: German**

```
* $r\ a\ d$  *
  $z$  

  $v$

  $d$

  $r\ a\ t\$  
```

References
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have $z\$ or $v\$ or $d\$ (where $\$ = word edge).

**Example: German**

```
* $ r a d $  
  *z$  
  *v$  
  *d$
```

```
$r a t$  
```

References
Example Without Tier: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don’t have \( z $ \) or \( v $ \) or \( d $ \) (where \( $ = \) word edge).

Example: German

\[
\begin{align*}
* s & \quad r & \quad a & \quad d & \\
* z & \quad v & \quad d & \\
$ r & \quad a & \quad t & \\
\end{align*}
\]
Example Without Tier: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose:**
  - $[-\text{voice}] = \{s,f\}$
  - $V = \{a,i,u\}$
- **Then:** don’t have $\text{asa, a}f\text{a, asi, a}f\text{i, ...}$

Example

```
* $a z u s a$
```
Example Without Tier: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- Suppose:
  - $[\neg \text{voice}] = \{s,f\}$
  - $V = \{a,i,u\}$
- Then: don’t have asa, aʃa, asi, aʃi, . . .

Example

* $a z u s a$
Example Without Tier: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose:**
  - $[-\text{voice}] = \{s, f\}$
  - $V = \{a, i, u\}$
- **Then:** don’t have asa, aʃa, asi, aʃi, . . .

Example

* $a z u s a$
Example Without Tier: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose:**
  - \([-\text{voice}] = \{s, \text{ʃ}\}\]
  - \(V = \{a, i, u\}\)
- **Then:** don’t have \(asa, aʃa, asi, aʃi, \ldots\)

Example

\[
* $a z u s a$
\]
Example Without Tier: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose:**
  - \([-\text{voice}] = \{s,f\}\]
  - \(V = \{a,i,u\}\)
- **Then:** don’t have \(asa, asa, asi, asi, \ldots\)

Example

\*$ a z u s a $*
Example Without Tier: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- Suppose:
  - \([-\text{voice}] = \{s, f\}\)
  - \(V = \{a, i, u\}\)
- Then: don’t have \(asa, asa, asi, asi, \ldots\)

Example

\[ * \; a \; z \; u \; s \; a \; * \]
Adding Tiers: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (s,z) or −anterior (∫,ʒ).
- In other words: Don’t mix purple and teal.

\[
\begin{array}{cccc}
\ast s\& \ast z \\
\ast ∫ & \ast ʒ
\end{array}
\]

- But: Sibilants can be arbitrarily far away from each other!

Example: Samala

\[
\ast h a s x i n t i l a w a ∫ \$
\]

\[
$h a ∫ x i n t i l a w a ∫$
\]
Adding Tiers: Samala Sibilant Harmony

If multiple sibilants occur in the same word, they must all be $+$anterior ($s, z$) or $-$anterior ($\mathring{s}, \mathring{z}$).

In other words: Don’t mix purple and teal.

$\ast s\mathring{s} \ast s\mathring{z} \ast z\mathring{s} \ast z\mathring{z}$

But: Sibilants can be arbitrarily far away from each other!

Example: Samala

$\ast \$ h a s x i n t i l a w a \mathring{f} \$$

$\$ h a \mathring{f} x i n t i l a w a \mathring{f} \$
Adding Tiers: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be $+$ anterior ($s,z$) or $-$ anterior ($ʃ,ʒ$).
- In other words: Don’t mix purple and teal.

\*$sʃ*$ \*$sʒ*$ \*$zʃ*$ \*$zʒ*$ \*$ʃs*$ \*$ʒs*$ \*$ʃz*$ \*$ʒz*$

- But: Sibilants can be arbitrarily far away from each other!

**Example: Samala**

\*$ʃ x i n t i l a w a ʃ*$

\*$ʃ x i n t i l a w a ʃ*$

\*$s x i n t i l a w a ʃ*$

\*$s x i n t i l a w a ʃ*$
Adding Tiers: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (s,z) or −anterior (ʃ,ʒ).
- In other words: Don’t mix purple and teal.

\[
\begin{align*}
&*sʃ & *sʒ & *zʃ & *zʒ \\
&*ʃs & *ʒs & *ʒz & *zʒ
\end{align*}
\]

- But: Sibilants can be arbitrarily far away from each other!

Example: Samala

\[
\begin{align*}
* & $h a \mathbf{s} \times i n t i l a w a \mathbf{ʃ}$ \\
$ & $h a \mathbf{ʃ} \times i n t i l a w a \mathbf{ʃ}$
\]
Adding Tiers: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be $+$anterior ($s, z$) or $-$anterior ($ʃ, ʒ$).
- In other words: Don’t mix purple and teal.

$$\ast sʃ \ast sʒ \ast zʃ \ast zʒ$$

$$\astʃs \astʃʒ \astzs \astzs$$

- But: Sibilants can be arbitrarily far away from each other!

Example: Samala

$\ast \$\text{has} s x i n t i l a w a \ʃ \$\ast$

$\$\text{ha}ʃ x i n t i l a w a \ʃ \$\$

$\ast \$ s t a j a n o w o n w a \ʃ \$\ast$
Adding Tiers: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (s,z) or −anterior (ʃ,ʒ).
- In other words: Don’t mix purple and teal.

\[
*ʃʃ \quad *ʃʒ \quad *ʒʃ \quad *ʒʒ \\
*ʃs \quad *ʒs \quad *ʃz \quad *ʒz
\]

- But: Sibilants can be arbitrarily far away from each other!

Example: Samala

\[
*$h a s x i n t i l a w a ʃ*$ \\
*$ʃ x i n t i l a w a ʃ*$ \\
*$ʃ a n o w o n w a ʃ*
\]
Making Long-Distance Dependencies Local

Let's take a hint from phonology: create locality with a tier. (Heinz et al. 2011)

**Restriction 1:** only 1 tier

**Restriction 2:** projection is determined by the segments, not their environment

---

**Example: Samala Revisited**

1. Project sibilant tier


*$h a s x ʃ i n t i l a w a ʃ*$  
*$h a ʃ x i n t i l a w a ʃ*$
Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

**Example: Samala Revisited**

1. Project sibilant tier

2. $*s\exists, *s_3, *z\exists, *z_3, *s, *s_3, *z, *z_3$

*$h a s x i n t i l a w a \exists*$  
*$h a \exists x i n t i l a w a \exists*$
Making Long-Distance Dependencies Local

- Let's take a hint from phonology:
  create locality with a tier.
  (Heinz et al. 2011)

- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

### Example: Samala Revisited

1. Project sibilant tier

2. \*sʃ, \*sʃ, \*zʃ, \*zʃ, \*ʃs, \*ʒs, \*ʃz, \*ʒz

```
  $ s ʃ$
  |    | ʃ $
  *$ h a ʃ x i n t i l a w a $
  $ h a ʃ x i n t i l a w a $
```

Jeff Heinz
Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a tier. (Heinz et al. 2011)
- **Restriction 1**: only 1 tier
- **Restriction 2**: projection is determined by the segments, not their environment

### Example: Samala Revisited

1. Project sibilant tier

\[
\begin{array}{cccc}
\$ & s & \mathcal{J} & \mathcal{J} \\
| & | & | & | \\
^*\$ & h & a & s & x & i & n & t & i & l & a & w & a & \mathcal{J} & \$ \\
\end{array}
\]

\[
\begin{array}{cccc}
\$ & \mathcal{J} & \mathcal{J} & \mathcal{J} \\
| & | & | & | \\
\$ & h & a & \mathcal{J} & x & i & n & t & i & l & a & w & a & \mathcal{J} & \$ \\
\end{array}
\]
Making Long-Distance Dependencies Local

Let's take a hint from phonology: create locality with a **tier**.
(Heinz et al. 2011)

**Restriction 1:** only 1 tier

**Restriction 2:** projection is determined by the segments, not their environment

---

**Example: Samala Revisited**

1. Project sibilant tier

2. $s$, $s3$, $z$, $z3$, $s$, $3s$, $z$, $3z$

```
$ s   
  |  
* $ has xintilawa $  
  |  
$ h a $  
  |  
$  
  |  
$  
  |  
```

---

Jeff Heinz
Making Long-Distance Dependencies Local

Let's take a hint from phonology: create locality with a tier. (Heinz et al. 2011)

- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

**Example: Samala Revisited**

1. Project sibilant tier
2. $^*s\int, ^*s_3, ^*z\int, ^*z_3, ^*s, ^*_3s, ^*_3z, ^*_3z$

```
$ s \int s $ $ \int $ $ \int $

* $ h a s x i n t i l a w a \int $ $ h a \int x i n t i l a w a \int $
```
Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**.  
  (Heinz et al. 2011)

- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

### Example: Samala Revisited

1. Project sibilant tier

2. $^{*}\text{s}}$, $^{*}\text{s}_3$, $^{*}\text{z}}$, $^{*}\text{z}_3$, $^{*}\text{s}$, $^{*}\text{s}_3$, $^{*}\text{z}$, $^{*}\text{z}_3$

\[
\begin{align*}
\text{Project sibilant tier: } & \quad \text{S} \quad \text{S} \\
\text{Samala: } & \quad *\text{S} \quad \text{S} \\
\end{align*}
\]
Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a tier. (Heinz et al. 2011)

- **Restriction 1**: only 1 tier

- **Restriction 2**: projection is determined by the segments, not their environment

---

**Example: Samala Revisited**

1. Project sibilant tier

2. $^*s\$, $^*s_3\$, $^*z\$, $^*z_3\$, $^*_s\$, $^*_s_3\$, $^*_z\$, $^*_z_3\$

   ```
   $s$  \[\text{\_\_\_}\]  $\int$ $\int$ $\int$ $\int$
   $^*$$h\ a\ s\ x\ i\ n\ t\ i\ l\ a\ w\ a\ \int$
   
   $^*$$h\ a\ \int\ x\ i\ n\ t\ i\ l\ a\ w\ a\ \int$
   ```
Making Long-Distance Dependencies Local

- Let’s take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1**: only 1 tier
- **Restriction 2**: projection is determined by the segments, not their environment

**Example: Samala Revisited**

1. Project sibilant tier
2. \[*s\], \[*s\_3\], \[*z\], \[*z\_3\], \[*s\], \[*s\_3\], \[*z\], \[*z\_3\]

\[\begin{array}{c}
\text{has} \quad \text{xintila} \quad \text{wa} \\
\end{array}\]

\[\begin{array}{c}
\text{ha} \quad \text{xintila} \quad \text{wa} \\
\end{array}\]
Culminativity: Simple Counting with TSL

**Culminativity** phonological word contains exactly 1 stress

**Example**

1. Project stress tier
2. $$++, **VV

* $b a n a n a$

$ b a n a n a$

$ b a n a n a$
Culminativity: Simple Counting with TSL

**Culminativity** phonological word contains exactly 1 stress

**Example**

1. Project stress tier
2. $**$, $∗ÝÝ$

* $b a n ã n a$  $b ã n ã n a$  $b a n a n a$
Culminativity: Simple Counting with TSL

**Culminativity** phonological word contains exactly 1 stress

**Example**

1. Project stress tier

2. *$$, *VV

```
$ á $
|   |   |
* $ b a n á n a $  $ b á n á n a $  $ b a n a n a $
```
Culminativity: Simple Counting with TSL

**Culminativity** phonological word contains exactly 1 stress

**Example**

1. Project stress tier
2. *$$, *́V́V*

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>á</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>b</td>
<td>a</td>
<td>n</td>
<td>á</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>á</td>
<td>n</td>
<td>á</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>b</td>
<td>a</td>
<td>n</td>
<td>á</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>á</td>
<td>n</td>
<td>á</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>n</td>
<td>a</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Culminativity:** phonological word contains exactly 1 stress

### Example

1. Project stress tier
2. *$$*, *´VV*

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>á</td>
<td>$</td>
<td>$</td>
<td>á</td>
<td>á</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*$ b a n á n a $</td>
<td>$ b á n á n a $</td>
<td>$ b a n a n a $</td>
<td>$ b a n a n a $</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why is TSL Interesting?

- Linguistically natural
- Correct and very efficient learning algorithm (Jardine and McMullin 2017)
- Low resource demands $\Rightarrow$ cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate many unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

$hashxintilawaf*$ $stajanowonwaf*$
Why is TSL Interesting?

- Linguistically natural
- Correct and very efficient learning algorithm
  (Jardine and McMullin 2017)
- Low resource demands $\Rightarrow$ cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate many unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

$\text{*s*} \text{h a s} x \text{i n t i l a w a s*} \text{j}$

$\text{*s*} \text{t a j a n o w o n w a s*}$
Why is TSL Interesting?

- Linguistically natural
- Correct and very efficient learning algorithm (Jardine and McMullin 2017)
- Low resource demands ⇒ cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate many unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

```
$ s $                $ s t a j a n o w o n w a $  
|                 |
$ h a s x i n t i l a w a $  *
```
Why is TSL Interesting?

▶ Linguistically natural
▶ Correct and very efficient learning algorithm (Jardine and McMullin 2017)
▶ Low resource demands ⇒ cognitively plausible
▶ Captures wide range of phonotactic dependencies
▶ Cannot generate many unattested patterns

Example: First-Last Harmony

▶ Harmony only holds between initial and final segments
▶ Linguistically plausible, yet unattested

\[
\begin{align*}
\text{Has} & \quad \text{ints} & \quad \text{law} & \quad \text{st} & \quad \text{ja} & \quad \text{n} & \quad \text{ow} & \quad \text{on} & \quad \text{wa} & \\
\text{sh} & & & & & & & & & \\
\end{align*}
\]
TSL Across Language Modules

- c-command
- Merge & Move
- Syntax
- Phonotactics
- Morphotactics
- first-last harmony
- non-final RHOL
- UTP
- Lowering
- unbounded reduplication
- unbounded circumfixation
TSL Semantics

- TSL seems to play an important role in
  - phonology,
  - morphology,
  - syntax.
- What’s missing? Semantics!
- But TSL is about strings/trees.
- What is a semantic string language?
Formal Language Theory for Semantics

1. **Quantifier Languages**
   Meanings as strings of truth values
   (van Benthem 1986)

2. **“Tense Languages”**
   Meanings as strings of events
   (Fernando 2011)

- I’ll only talk about quantifier languages here.
- Ongoing work with Rob Pasternak on subregularity of tense languages
Evaluating the Truth of Quantifiers

(1) a. Every student cheated.
   b. No student cheated.
   c. Some student cheated.
   d. Three students cheated.

<table>
<thead>
<tr>
<th>students</th>
<th>John</th>
<th>Mary</th>
<th>Sue</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheated</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

| string | Y   | N   | Y   |

- (1a): False, because the string contains a N
- (1b): False, because the string contains a Y
- (1c): True, because the string contains a Y
- (1d): False, because the string does not contain three Ys
Evaluating the Truth of Quantifiers

(1) a. Every student cheated.
b. No student cheated.
c. Some student cheated.
d. Three students cheated.

<table>
<thead>
<tr>
<th>students</th>
<th>John</th>
<th>Mary</th>
<th>Sue</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheated</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

| string | Y | N | Y |

- (1a): False, because the string contains a N
- (1b): False, because the string contains a Y
- (1c): True, because the string contains a Y
- (1d): False, because the string does not contain three Ys
Formalization Step 1: Binary String Languages

**Idea**: Convert relation between sets $A$ and $B$ into set of Yes/No-strings

**Definition (Binary String Language)**

1. $A, B$: arbitrary sets
2. $f(A, B)$: maps each $a \in A$ to $Y$ if $a \in B$, otherwise $N$
3. $e(A)$: arbitrary enumeration of $A$
4. $L(A, B)$: all $e(A)$, relabeled by $f(A, B)$
Example

1 Set of students: {John, Mary, Sue}
Set of cheaters: {John, Sue, Bill, Peter}

2 $f(A, B)$:
- John $\mapsto$ Y
- Mary $\mapsto$ N
- Sue $\mapsto$ Y

3 $e(A)$:
- 1) John Mary Sue
- 2) John Sue Mary
- 3) Mary John Sue
- 4) Mary Sue John
- 5) Sue John Mary
- 6) Sue Mary John

4 $L(A, B)$:
\[
\begin{cases}
  \text{YNY,} \\
  \text{YYN,} \\
  \text{NYY}
\end{cases}
\]
Formalization Step 2: Quantifier Language

**Idea:** Every quantifier is a set of acceptable Yes/No-strings

**Definition (Quantifier Language)**

$L(Q)$ is the **quantifier language** of $Q$ iff it holds for all $A$ and $B$ that $Q(A, B)$ is true iff $L(A, B) \subseteq L(Q)$.

**Example**

- $L(\text{every}) = \text{set of all strings containing no } N$
- **Why?**
  - $\text{every}(A, B)$ iff $A \subseteq B$
  - If $A \subseteq B$, then no binary string contains $N$.
  - If some binary string contains $N$, then $A \nsubseteq B$. 
Formalization Step 2: Quantifier Language

**Idea:** Every quantifier is a set of acceptable Yes/No-strings

**Definition (Quantifier Language)**

$L(Q)$ is the *quantifier language* of $Q$ iff it holds for all $A$ and $B$ that $Q(A, B)$ is true iff $L(A, B) \subseteq L(Q)$.

**Example**

- $L(\text{every}) = \text{set of all strings containing no N}$
- Why?
  - $\text{every}(A, B)$ iff $A \subseteq B$
  - If $A \subseteq B$, then no binary string contains N.
  - If some binary string contains N, then $A \not\subseteq B$. 
# A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$Y</td>
</tr>
<tr>
<td>no</td>
<td>$N</td>
</tr>
<tr>
<td>some</td>
<td>$Y</td>
</tr>
<tr>
<td>at least $n$</td>
<td>$Y</td>
</tr>
<tr>
<td>at most $n$</td>
<td>$Y</td>
</tr>
<tr>
<td>exactly $n$</td>
<td>$Y</td>
</tr>
<tr>
<td>not all</td>
<td>$N</td>
</tr>
<tr>
<td>all but $n$</td>
<td>$N</td>
</tr>
<tr>
<td>most</td>
<td>$Y</td>
</tr>
<tr>
<td>an even number</td>
<td>$Y</td>
</tr>
</tbody>
</table>
A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td></td>
</tr>
<tr>
<td>some</td>
<td></td>
</tr>
<tr>
<td>at least $n$</td>
<td></td>
</tr>
<tr>
<td>at most $n$</td>
<td></td>
</tr>
<tr>
<td>exactly $n$</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td></td>
</tr>
<tr>
<td>all but $n$</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
## A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
</tr>
<tr>
<td>some</td>
<td></td>
</tr>
<tr>
<td>at least $n$</td>
<td></td>
</tr>
<tr>
<td>at most $n$</td>
<td></td>
</tr>
<tr>
<td>exactly $n$</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td></td>
</tr>
<tr>
<td>all but $n$</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
# A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
</tr>
<tr>
<td>at least $n$</td>
<td></td>
</tr>
<tr>
<td>at most $n$</td>
<td></td>
</tr>
<tr>
<td>exactly $n$</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td></td>
</tr>
<tr>
<td>all but $n$</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
## A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>(</td>
</tr>
<tr>
<td>no</td>
<td>(</td>
</tr>
<tr>
<td>some</td>
<td>(</td>
</tr>
<tr>
<td>at least (n)</td>
<td>(</td>
</tr>
<tr>
<td>at most (n)</td>
<td></td>
</tr>
<tr>
<td>exactly (n)</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td></td>
</tr>
<tr>
<td>all but (n)</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
## A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
</tr>
<tr>
<td>at least $n$</td>
<td>$</td>
</tr>
<tr>
<td>at most $n$</td>
<td>$</td>
</tr>
<tr>
<td>exactly $n$</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td></td>
</tr>
<tr>
<td>all but $n$</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td></td>
</tr>
<tr>
<td>some</td>
<td></td>
</tr>
<tr>
<td>at least n</td>
<td></td>
</tr>
<tr>
<td>at most n</td>
<td></td>
</tr>
<tr>
<td>exactly n</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td></td>
</tr>
<tr>
<td>all but n</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
## A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
</tr>
<tr>
<td>at least $n$</td>
<td>$</td>
</tr>
<tr>
<td>at most $n$</td>
<td>$</td>
</tr>
<tr>
<td>exactly $n$</td>
<td>$</td>
</tr>
<tr>
<td>not all</td>
<td>$</td>
</tr>
<tr>
<td>all but $n$</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
# A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
</tr>
<tr>
<td>at least n</td>
<td>$</td>
</tr>
<tr>
<td>at most n</td>
<td>$</td>
</tr>
<tr>
<td>exactly n</td>
<td>$</td>
</tr>
<tr>
<td>not all</td>
<td>$</td>
</tr>
<tr>
<td>all but n</td>
<td>$</td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>an even number</td>
<td></td>
</tr>
</tbody>
</table>
## A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>(</td>
</tr>
<tr>
<td>no</td>
<td>(</td>
</tr>
<tr>
<td>some</td>
<td>(</td>
</tr>
<tr>
<td>at least (n)</td>
<td>(</td>
</tr>
<tr>
<td>at most (n)</td>
<td>(</td>
</tr>
<tr>
<td>exactly (n)</td>
<td>(</td>
</tr>
<tr>
<td>not all</td>
<td>(</td>
</tr>
<tr>
<td>all but (n)</td>
<td>(</td>
</tr>
<tr>
<td>most</td>
<td>(</td>
</tr>
</tbody>
</table>

an even number
A Sample of Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
</tr>
<tr>
<td>at least $n$</td>
<td>$</td>
</tr>
<tr>
<td>at most $n$</td>
<td>$</td>
</tr>
<tr>
<td>exactly $n$</td>
<td>$</td>
</tr>
<tr>
<td>not all</td>
<td>$</td>
</tr>
<tr>
<td>all but $n$</td>
<td>$</td>
</tr>
<tr>
<td>most</td>
<td>$</td>
</tr>
<tr>
<td>an even number</td>
<td>$</td>
</tr>
</tbody>
</table>
# TSL Quantifier Languages for every and no

**every** is TSL Without Tiers

| *N  | $YNYNY$ | $YYNY$ |

**no** is TSL Without Tiers

| *Y  | $NNNNN$ | $NNYN$ |
TSL Quantifier Languages for every and no

every is TSL Without Tiers

* N

$Y Y Y Y Y$ $Y Y N Y$

no is TSL Without Tiers

* Y

$N N N N$ $N N Y N$
TSL Quantifier Languages for every and no

**every** is TSL Without Tiers

\*[N

\[\begin{array}{cccc}
Y & Y & Y & Y \\
\end{array}\] \[\begin{array}{cccc}
Y & Y & N & Y \\
\end{array}\]

**no** is TSL Without Tiers

\*[\[Y

\[\begin{array}{cccc}
N & N & N & N \\
\end{array}\] \[\begin{array}{cccc}
N & N & Y & N \\
\end{array}\]
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

\[ *N \]

\[ \begin{array}{c}
\$ Y Y Y Y Y \$ \\
\$ Y Y N Y Y \$
\end{array} \]

**no** is TSL Without Tiers

\[ *Y \]

\[ \begin{array}{c}
\$ N N N N N \$ \\
\$ N N Y N N \$
\end{array} \]
# TSL Quantifier Languages for *every* and *no*

### *every* is TSL Without Tiers

| *N | $Y$ | $Y$ | $Y$ | $|$ |
|----|-----|-----|-----|----|
| $|$ | $Y$ | $Y$ | $|$ | $N$ |
| $|$ | $|$ | $|$ | $|$ | $N$ |

### *no* is TSL Without Tiers

| *Y | $N$ | $N$ | $N$ | $N$ | $|$ |
|----|-----|-----|-----|-----|----|
| $|$ | $N$ | $N$ | $|$ | $Y$ | $N$ |
| $|$ | $|$ | $|$ | $|$ | $|$ | $N$ |
TSL Quantifier Languages for *every* and *no*

### every is TSL Without Tiers

| *N |  
|---|---
| $\text{Y Y Y Y Y}$ | $\text{Y Y Y N Y}$ |

### no is TSL Without Tiers

<table>
<thead>
<tr>
<th>*Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{N N N N N}$</td>
</tr>
</tbody>
</table>
# TSL Quantifier Languages for every and no

**every** is TSL Without Tiers

| *N |  
$\text{Y Y Y Y Y}$ | $\text{Y Y N Y}$ |

**no** is TSL Without Tiers

| *Y |  
$\text{N N N N N}$ | $\text{N N Y N}$ |
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

| *N | $Y Y Y Y Y$ | $Y Y N Y$ |

**no** is TSL Without Tiers

| *Y | $N N N N N$ | $N N Y N$ |
TSL Quantifier Languages for *every* and *no*

<table>
<thead>
<tr>
<th>every is TSL Without Tiers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>N</em></td>
<td></td>
</tr>
<tr>
<td>$Y Y Y Y Y$</td>
<td>$Y Y N Y$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no is TSL Without Tiers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Y</em></td>
<td></td>
</tr>
<tr>
<td>$N N N N N$</td>
<td>$N N Y N$</td>
</tr>
</tbody>
</table>
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

* N

$ Y Y Y Y Y $  $ Y Y N Y Y $  

**no** is TSL Without Tiers

* Y

$ N N N N N $  $ N N Y N N $
TSL Quantifier Languages for every and no

**every** is TSL Without Tiers

*\( N \)

\[
\begin{array}{c}
\$ Y Y Y Y Y \$
\end{array}
\]

\[
\begin{array}{c}
\$ Y Y Y N Y \$
\end{array}
\]

**no** is TSL Without Tiers

*\( Y \)

\[
\begin{array}{c}
\$ N N N N N \$
\end{array}
\]

\[
\begin{array}{c}
\$ N N Y N N \$
\end{array}
\]
### TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

\[ \ast N \]

\[
\begin{array}{c}
\$ \ Y \ Y \ Y \ Y \ Y \ $ \\
\$ \ Y \ Y \ Y \ N \ Y \ $ \\
\end{array}
\]

**no** is TSL Without Tiers

\[ \ast Y \]

\[
\begin{array}{c}
\$ \ N \ N \ N \ N \ N \ $ \\
\$ \ N \ N \ Y \ Y \ N \ $ \\
\end{array}
\]
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

| *N | $Y\ Y\ Y\ Y\ Y$ | $Y\ Y\ N\ Y$ |

**no** is TSL Without Tiers

| *Y | $N\ N\ N\ N\ N$ | $N\ N\ Y\ N\ N$ |
## TSL Quantifier Languages for *every* and *no*

<table>
<thead>
<tr>
<th>every is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>*N</td>
</tr>
<tr>
<td>$ Y Y Y Y Y $</td>
</tr>
<tr>
<td>$ Y Y N Y $</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Y</td>
</tr>
<tr>
<td>$ N N N N $</td>
</tr>
<tr>
<td>$ N N Y N $</td>
</tr>
</tbody>
</table>
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

\[ *N \]

\[ \begin{array}{ll}
    \$ Y Y Y Y Y \$ & \$ Y Y N Y \$ \\
\end{array} \]

**no** is TSL Without Tiers

\[ *Y \]

\[ \begin{array}{ll}
    \$ N N N N N \$ & \$ N N Y N N \$ \\
\end{array} \]
### TSL Quantifier Languages for *every* and *no*

#### **every** is TSL Without Tiers

* \( \ast N \)  

\[
\begin{array}{cccc}
\$ & Y & Y & Y & Y & \$ \\
\$ & Y & Y & N & Y & \$
\end{array}
\]

#### **no** is TSL Without Tiers

* \( \ast Y \)  

\[
\begin{array}{cccc}
\$ & N & N & N & N & \$ \\
\$ & N & N & Y & N & \$
\end{array}
\]
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

| *N | $Y Y Y Y Y$ | $Y Y N Y$ |

**no** is TSL Without Tiers

| *Y | $N N N N$ | $N N Y N N$ |
TSL Quantifier Languages for *every* and *no*

**every** is TSL Without Tiers

\[ *N \]

\[ \begin{array}{c}
\$ YYYYY \$ \\
\$ YYYNYY \$
\end{array} \]

**no** is TSL Without Tiers

\[ *Y \]

\[ \begin{array}{c}
\$ NNNNN \$ \\
\$ NNNYN \$
\end{array} \]
### TSL Quantifier Languages for *every* and *no*

<table>
<thead>
<tr>
<th>every is TSL Without Tiers</th>
<th>no is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>N</em></td>
<td><em>Y</em></td>
</tr>
<tr>
<td>$ Y Y Y Y Y $</td>
<td>$ Y Y N Y $</td>
</tr>
<tr>
<td>$ N N N N N $</td>
<td>$ N N Y N $</td>
</tr>
</tbody>
</table>
## TSL Quantifier Languages for *every* and *no*

### every is TSL Without Tiers

<table>
<thead>
<tr>
<th>*N</th>
<th>$ Y Y Y Y Y $</th>
<th>$ Y Y N Y $</th>
</tr>
</thead>
</table>

### no is TSL Without Tiers

<table>
<thead>
<tr>
<th>*Y</th>
<th>$ N N N N N $</th>
<th>$ N N Y N $</th>
</tr>
</thead>
</table>
TSL Quantifier Languages for *every* and *no*

<table>
<thead>
<tr>
<th>every is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>N</em></td>
</tr>
<tr>
<td>$ Y Y Y Y Y $</td>
</tr>
<tr>
<td>$ Y Y N Y $</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Y</em></td>
</tr>
<tr>
<td>$ N N N N N $</td>
</tr>
<tr>
<td>$ N N Y N $</td>
</tr>
</tbody>
</table>
TSL Quantifier Languages for *every* and *no*

<table>
<thead>
<tr>
<th>every is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>*N</td>
</tr>
<tr>
<td>$ Y Y Y Y Y $</td>
</tr>
<tr>
<td>$ Y Y N Y $</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no is TSL Without Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Y</td>
</tr>
<tr>
<td>$ N N N N N $</td>
</tr>
<tr>
<td>$ N N Y N $</td>
</tr>
</tbody>
</table>
Most Quantifier Languages Require a Tier

**some** is TSL With Tier \{Y\}

\*$\$$  

\[ N \ N \ Y \ N \ ] \quad \[ N \ N \ N \ N \ ]

**all but 3** is TSL With Tier \{N\}

\*$\$$, \*$N\$$, \*$NN\$$, \*$NNNN$

\[ N \ N \ N \ Y \ N \ ] \quad \[ N \ Y \ Y \ Y \ N \ ] \quad \[ N \ N \ N \ N \ ]
Most Quantifier Languages Require a Tier

**some** is TSL With Tier \{Y\}

\[*\$

\[
\begin{array}{ccc}
  \$ & Y & \$
  \\
  \mid & \mid & \mid \\
  \$ N N Y N \$ & & \$ N N N N N \$
\end{array}
\]

**all but 3** is TSL With Tier \{N\}

\[*\$, *\$N\$, *\$NN\$, *\$NNNN\$

\[
\begin{array}{cc}
  \$ N N Y N \$ & \$ N Y Y N \$ & \$ N N N N N \$
\end{array}
\]
Most Quantifier Languages Require a Tier

<table>
<thead>
<tr>
<th>some is TSL With Tier {Y}</th>
</tr>
</thead>
<tbody>
<tr>
<td>*$$</td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$ \quad \text{Y} \quad $</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$ \quad \text{N} \quad \text{N} \quad \text{Y} \quad \text{N} \quad $</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$ \quad \text{N} \quad \text{N} \quad \text{N} \quad \text{N} \quad \text{N} \quad $</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>all but 3 is TSL With Tier {N}</th>
</tr>
</thead>
<tbody>
<tr>
<td>*$$, <em>$N$</em>, <em>$NN$</em>, <em>$NNNN$</em></td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$ \quad \text{N} \quad \text{N} \quad \text{Y} \quad \text{N} \quad $</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$ \quad \text{N} \quad \text{Y} \quad \text{Y} \quad \text{N} \quad $</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
<tr>
<td>[ \begin{array}{ccc}</td>
</tr>
<tr>
<td>$ \quad \text{N} \quad \text{N} \quad \text{N} \quad \text{N} \quad \text{N} \quad $</td>
</tr>
<tr>
<td>\end{array} ]</td>
</tr>
</tbody>
</table>
Most Quantifier Languages Require a Tier

some is TSL With Tier \{Y\}

\*\*$

\$ N N Y N \$

\$ Y \$

\$ N N N N N \$

all but 3 is TSL With Tier \{N\}

\*\*$, \*$N\*$, \*$NN\*$, \*$NNNN\*$
Most Quantifier Languages Require a Tier

**some** is TSL With Tier \( \{Y\} \)

$$
\text{**some** is TSL With Tier \( \{Y\} \)}

$$

all but 3 is TSL With Tier \( \{N\} \)

$$
\text{all but 3 is TSL With Tier \( \{N\} \)}

$$
Most Quantifier Languages Require a Tier

<table>
<thead>
<tr>
<th>some is TSL With Tier {Y}</th>
</tr>
</thead>
<tbody>
<tr>
<td>*$*$</td>
</tr>
<tr>
<td>$ Y $</td>
</tr>
<tr>
<td>$ N N Y N $</td>
</tr>
<tr>
<td>$ N N N $</td>
</tr>
</tbody>
</table>

\[ \text{all but 3 is TSL With Tier \{N\}} \]

\[ \*$*$, \*$N*$, \*$NN*$, \*$NNNN$ \]

\[ \$ N N Y N \$ \]
\[ \$ N Y Y N \$ \]
\[ \$ N N N N \$ \]
Most Quantifier Languages Require a Tier

**some is TSL With Tier \{Y\}**

\*\*$\\ N\\ N\\ Y\\ N\\ $\\ \$

\*\*$\\ Y\\ $\\ \$

\*\*$\\ $\\ $\\ \$

**all but 3 is TSL With Tier \{N\}**

\*\*$, \*\*$N*$, \*\*$NN*$, \*\*$NNNN$

\$\\ N\\ N\\ Y\\ N\\ $\\$

\$\\ N\\ Y\\ Y\\ N\\ $\\$

\$\\ N\\ N\\ N\\ N\\ N\\ $\\$
Most Quantifier Languages Require a Tier

\textbf{some} is TSL With Tier \{Y\}

\begin{align*}
\ast\ast\ast & \\
& \begin{array}{ccc}
\$ & Y & \$ \\
\$ & N & N \\
\$ & N & Y & N \\
\end{array} & \begin{array}{ccc}
\$ & \$ \\
\$ & N & N \\
\$ & N & N & N \\
\end{array}
\end{align*}

\textbf{all but 3} is TSL With Tier \{N\}

\begin{align*}
\ast\ast\ast, \ast\ast\ast N, \ast\ast\ast NN, \ast\ast\ast NNNN & \\
& \begin{array}{ccc}
\$ & N & N \\
\$ & N & N \\
\$ & N & Y & N \\
\end{array} & \begin{array}{ccc}
\$ & N & \$ \\
\$ & N & Y & Y & N \\
\$ & N & N & N & N \\
\end{array}
\end{align*}
Most Quantifier Languages Require a Tier

<table>
<thead>
<tr>
<th>Some is TSL With Tier {Y}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>*$*$</strong></td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All but 3 is TSL With Tier {N}</th>
</tr>
</thead>
<tbody>
<tr>
<td>***$*$, *<em>*$N</em>$, <em><em>*$NN</em>$, <em><em>*$NNNN</em>$</em></em></td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$</td>
</tr>
<tr>
<td>$</td>
</tr>
</tbody>
</table>
Most Quantifier Languages Require a Tier

**Some** is TSL With Tier \{Y\}

\*\*\*

\begin{array}{ccc}
\$ & Y & \$\\
\$ N & N & Y & N & \$\\
\end{array}

\begin{array}{ccc}
\$ & $ & $ & $ & $ \\
\end{array}

\begin{array}{ccc}
\end{array}

\begin{array}{ccc}
\$ N & N & N & N & \$\\
\end{array}

**All but 3** is TSL With Tier \{N\}

\*\*\*, \*\$N\$, \*\$NN\$, \*\$NNNN\$

\begin{array}{ccc}
\$ N & N & N & \$\\
\$ N & \$\\
\$ N & N & Y & N & \$\\
\end{array}

\begin{array}{ccc}
\end{array}

\begin{array}{ccc}
\$ N & N & N & N & \$\\
\$ N & N & N & N & \$\\
\end{array}

\begin{array}{ccc}
\$ N & N & N & N & \$\\
\$ N & N & N & N & \$\\
\$ N & N & N & N & \$\\
\end{array}
Most Quantifier Languages Require a Tier

### some is TSL With Tier \{Y\}

<table>
<thead>
<tr>
<th></th>
<th>$N$</th>
<th>$Y$</th>
<th>$N$</th>
<th>$Y$</th>
<th>$N$</th>
<th>$Y$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### all but 3 is TSL With Tier \{N\}

<table>
<thead>
<tr>
<th></th>
<th>$N$</th>
<th>$N$</th>
<th>$N$</th>
<th>$N$</th>
<th>$N$</th>
<th>$N$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\*$*$, \*$N*$, \*$NN*$, \*$NNNN*$
Most Quantifier Languages Require a Tier

**some** is TSL With Tier \{Y\}

```latex
*\$$

\begin{array}{llll}
\$ & \mathbf{Y} & \$ & \\
\$ & \mathbf{N} & \mathbf{N} & \mathbf{Y} & \mathbf{N} & \$
\end{array}
\begin{array}{llll}
\$ & \$ & \\
\$ & \mathbf{N} & \mathbf{N} & \mathbf{N} & \mathbf{N} & \$
\end{array}
```

**all but 3** is TSL With Tier \{N\}

```latex
*\$$, *\$$\text{N}\$$, *\$$\text{NN}\$$, *\$$\text{NNNN}\$

\begin{array}{llll}
\$ & \mathbf{N} & \mathbf{N} & \mathbf{N} & \$
\end{array}
\begin{array}{llll}
\$ & \mathbf{N} & \mathbf{N} & \$
\end{array}
\begin{array}{llll}
\$ & \mathbf{N} & \mathbf{N} & \mathbf{N} & \mathbf{N} & \$
\end{array}
\begin{array}{llll}
\$ & \mathbf{N} & \mathbf{N} & \mathbf{N} & \mathbf{N} & \$
\end{array}
```

19
Most Quantifier Languages Require a Tier

**some** is TSL With Tier \{Y\}

```latex
\*
$$
\begin{array}{c|c|c|c|c|}
\$ & \text{Y} & \$ & \$ & \$ \\
\| & \| & \| & \| & \\
\$ & \text{N} & \text{N} & \text{Y} & \text{N} \\
\$ & \text{N} & \text{N} & \text{N} & \text{N} \\
\end{array}
$$
```

**all but 3** is TSL With Tier \{N\}

```latex
\*
\*N\*, \*NN\*, \*NNNN\*

\begin{array}{c|c|c|c|c|}
\$ & \text{N} & \text{N} & \text{N} & \$ \\
\| & \| & \| & \| & \\
\$ & \text{N} & \text{N} & \text{Y} & \text{N} \\
\$ & \text{N} & \text{N} & \text{N} & \text{N} \\
\$ & \text{N} & \text{N} & \text{N} & \text{N} \\
\$ & \text{N} & \text{N} & \text{N} & \text{N} \\
\end{array}
```

19
Most Quantifier Languages Require a Tier

<table>
<thead>
<tr>
<th>some is TSL With Tier {Y}</th>
</tr>
</thead>
<tbody>
<tr>
<td>*$$</td>
</tr>
<tr>
<td>$ \ Y \ $</td>
</tr>
<tr>
<td>$ \ N \ N \ Y \ N \ $</td>
</tr>
<tr>
<td>$ \ N \ N \ N \ N \ $</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>all but 3 is TSL With Tier {N}</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ \ N \ N \ N $$</td>
</tr>
<tr>
<td>$ \ N \ N \ Y \ N \ $</td>
</tr>
<tr>
<td>$ \ N \ N \ N \ N \ $</td>
</tr>
<tr>
<td>$ \ N \ N \ Y \ Y \ N \ $</td>
</tr>
<tr>
<td>$ \ N \ N \ N \ N \ $</td>
</tr>
</tbody>
</table>
Most Quantifier Languages Require a Tier

**some** is TSL With Tier \{Y\}

**all but 3** is TSL With Tier \{N\}
## TSL Descriptions for Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
<th>(n)-grams</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>(</td>
<td>N</td>
<td>= 0)</td>
</tr>
<tr>
<td>no</td>
<td>(</td>
<td>Y</td>
<td>= 0)</td>
</tr>
<tr>
<td>some</td>
<td>(</td>
<td>Y</td>
<td>\geq 1)</td>
</tr>
<tr>
<td>at least (n)</td>
<td>(</td>
<td>Y</td>
<td>\geq n)</td>
</tr>
<tr>
<td>at most (n)</td>
<td>(</td>
<td>Y</td>
<td>\leq n)</td>
</tr>
<tr>
<td>exactly (n)</td>
<td>(</td>
<td>Y</td>
<td>= n)</td>
</tr>
<tr>
<td>not all</td>
<td>(</td>
<td>N</td>
<td>\geq 1)</td>
</tr>
<tr>
<td>all but (n)</td>
<td>(</td>
<td>N</td>
<td>= n)</td>
</tr>
</tbody>
</table>
## Typology of Quantifiers

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
<th>(Paperno 2011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
<td></td>
</tr>
</tbody>
</table>
## Typology of Quantifiers

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>
## Typology of Quantifiers

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

(Paperno 2011)
Typology of Quantifiers

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

(Paperno 2011)
The Case of *most*

There is good semantic evidence that “most” is internally complex and hence **not monomorphemic**. (Hackl 2009)

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>
A New Upper Bound on Typological Variation

TSL Interpretation Conjecture

If a language uses a quantifier as a monomorphemic determiner, then its quantifier language must be TSL.
TSL is Too Large

- All monomorphemic quantifiers are TSL.
- But not all TSL-definable quantifiers are monomorphemic.
- Why might that be?

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
</tbody>
</table>
Monotonicity

Definition (Monotonicity)

- Let \( A \) and \( B \) be two sets with orders \( \leq_A \) and \( \leq_B \), respectively.
- A function \( f \) from \( A \) to \( B \) is **monotonic** iff
  \[
  x \leq_A y \implies f(x) \leq_B f(y)
  \]

- Monotonicity is similar to **No Crossing Branches** constraint.
Monotonicity

Definition (Monotonicity)

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is monotonic iff
  \[ x \leq_A y \Rightarrow f(x) \leq_B f(y) \]

- Monotonicity is similar to No Crossing Branches constraint.
Monotonicity

Definition (Monotonicity)

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is **monotonic** iff

$$x \leq_A y \Rightarrow f(x) \leq_B f(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.

```
  1   2   3
  |   |   |
 A   B   C
```
Monotonicity

Definition (Monotonicity)

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is monotonic iff

$$x \leq_A y \Rightarrow f(x) \leq_B f(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.
Monotonicity

Definition (Monotonicity)

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is **monotonic** iff

\[ x \leq_A y \Rightarrow f(x) \leq_B f(y) \]

- Monotonicity is similar to **No Crossing Branches** constraint.

![Diagram showing A, B, and C with branches between them](image-url)
Monotonicity

**Definition (Monotonicity)**

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is **monotonic** iff
  \[ x \leq_A y \Rightarrow f(x) \leq_B f(y) \]

- Monotonicity is similar to **No Crossing Branches** constraint.
Monotonicity

Definition (Monotonicity)

- Let \( A \) and \( B \) be two sets with orders \( \leq_A \) and \( \leq_B \), respectively.
- A function \( f \) from \( A \) to \( B \) is **monotonic** iff

\[
x \leq_A y \implies f(x) \leq_B f(y)
\]

- Monotonicity is similar to **No Crossing Branches** constraint.

```
1   2   3
|   |   |
A   B   C
```
Monotonicity

Definition (Monotonicity)

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is **monotonic** iff

$$x \leq_A y \Rightarrow f(x) \leq_B f(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.
Monotonicity

Definition (Monotonicity)

- Let $A$ and $B$ be two sets with orders $\leq_A$ and $\leq_B$, respectively.
- A function $f$ from $A$ to $B$ is monotonic iff

$$x \leq_A y \Rightarrow f(x) \leq_B f(y)$$

- Monotonicity is similar to No Crossing Branches constraint.

```
  1   2   3
 / \ /   /
A  B C
```
Monotonicity

Definition (Monotonicity)

- Let \( A \) and \( B \) be two sets with orders \( \leq_A \) and \( \leq_B \), respectively.
- A function \( f \) from \( A \) to \( B \) is **monotonic** iff
  \[
  x \leq_A y \implies f(x) \leq_B f(y)
  \]

- Monotonicity is similar to **No Crossing Branches** constraint.
Monotonicity in Language

- **Monotonicity in phonology**
  - No Crossing Branches constraint
  - Natural classes are convex

- **Monotonicity in morphology**
  - *ABA

- **Monotonicity in syntax**
  - Subcategorization $< A$-Move $< A'$-Move
  - Adjunct Island Constraint & Coordinate Structure Constraint

- **Monotonicity in semantics**
  - Everywhere...
Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

Project:

- Monotonicity forbids projecting only $N$. 

\[
\begin{array}{c|c|c|c}
Y & T & F \\
N &   &   \\
\end{array}
\]
Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

Project: Y

- Monotonicity forbids projecting only N.
Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

Project: $Y$ and $N$

- Monotonicity forbids projecting only $N$. 
Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

Project: nothing

- Monotonicity forbids projecting only $N$. 
Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

![Diagram](image)

- Project: forbidden

- Monotonicity forbids projecting only \( N \).
Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

\[
\begin{array}{c}
\text{Y} \\
\text{N} \\
\end{array}
\begin{array}{c}
\text{T} \\
\text{F} \\
\end{array}
\]

\text{Project:} \quad \text{forbidden}

- Monotonicity forbids projecting only \text{N}.
Adding Tiers for *every* and *no*

- *every* and *no* are the only quantifiers without tier
- **But:** no tier = tier containing everything

Example

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$ N $</td>
<td>$ N $</td>
<td>$ Y $</td>
<td>$ Y $</td>
<td>$ N $</td>
<td>$ N $</td>
<td>$ $</td>
<td></td>
</tr>
</tbody>
</table>

$ N $ $ N $ $ Y $ $ Y $ $ N $ $ N $ $ $

$ N $ $ N $ $ Y $ $ Y $ $ N $ $ N $ $ $

- So *every* and *no* can be viewed as using the tier \{Y,N\}.
- This satisfies monotonicity.
Remaining Issues & Extensions

- TSL also allows for some unnatural quantifiers; ruling them out requires some stipulations.
- What about fuzzy quantifiers? many, few, ...
- TSL makes cognitive complexity predictions; we’re working on experiments.
- Where else in semantics does TSL matter?
  - adverbial quantifiers
  - temporal semantics
  - modals
- But those are just small pieces of a much larger puzzle...
The Bigger Goal

- Computational approaches are abstract and content-neutral.
- This isn’t a problem but a virtue.
- Abstraction makes it possible to identify parallels between very different domains.

A Program of Subregular Unification

- To what extent can very different properties of language be reduced to the same computational property?
- What are the implications for
  - typological variation,
  - learnability,
  - cognition at large?
The Bigger Goal

- Computational approaches are abstract and content-neutral.
- This isn’t a problem but a **virtue**.
- Abstraction makes it possible to identify parallels between very different domains.

A Program of Subregular Unification

- To what extent can very different properties of language be reduced to the same computational property?
- What are the implications for
  - typological variation,
  - learnability,
  - cognition at large?
Place of Morphosemantics

- TSL
- Morphosyntax
- Phonotactics
- Morphotactics
- Syntax

- c-command
- Lowering
- Merge & Move
- monomorphemic quantifiers

- first-last harmony
- non-final RHOL
- UTP

- unbounded reduplication
- unbounded circumfixation
Conclusion

- Among determiners, all monomorphemic quantifiers have quantifier languages that are TSL.
- The opposite does not hold, additional restrictions on TSL are needed.
- Why does it matter? Because TSL is everywhere in language.
- Ultimate goal: computational explanation of typological variation
References I


References II


